REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-03-

0271

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 9 JULY 2003	3. REPORT TYPE AND DATES COVERED FINAL REPORT 15 APR 00 TO 14 APR 03

4. TITLE AND SUBTITLE EFFICIENT MODELING OF LARGE MOLECULES: GEOMETRY OPTIMIZATION, DYNAMICS AND CORRELATION ENERGY	5. FUNDING NUMBERS F49620-00-1-0281 3484/BS 61103D
6. AUTHOR(S) DR PETER PULAY AND DR JON BAKER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) UNIVERSITY OF ARKANSAS DEPARTMENT OF CHEMISTRY AND BIOCHEMISTRY CHEMISTRY 101 FAYETTEVILLE, AK 72701	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NL 4015 WILSON BLVD., ROOM 713 ARLINGTON, VA 22203-1954	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVE FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The original grant proposal comprised three parts, two of which were continuations of previous successful projects. The first project involves more efficient optimization techniques for very large molecules (containing several thousand atoms). The second is the development of algorithms for molecular dynamics in internal coordinates. The third project involves the efficient calculation of correlation energies for large (a few hundred atoms) molecules. This report summarizes our work in all three areas. Progress has been excellent throughout.

BEST AVAILABLE COPY

20030731 057

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLAS	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLAS	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLAS	20. LIMITATION OF ABSTRACT

FINAL REPORT TO THE AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

DEPSCoR/2000
Award No.: F49620-00-1-0281

Period: 1 August 2000 - 16 April 2003

Efficient Modeling of Large Molecules: Geometry Optimization, Dynamics and Correlation Energy

Peter Pulay and **Jon Baker**

Department of Chemistry and Biochemistry
University of Arkansas
Fayetteville
Arkansas 72701

Table of Content

INTRODUCTION

The original grant proposal comprised three parts, two of which were continuations of previous successful projects. The first project involves more efficient optimization techniques for very large molecules (containing several thousand atoms). The second is the development of algorithms for molecular dynamics in *internal* coordinates. The third project involves the efficient calculation of correlation energies for large (a few hundred atoms) molecules. This report summarizes our work in all three areas. Progress has been excellent throughout.

1. Efficient Geometry Optimization for Very Large Molecules

Geometry optimization of very large systems (e.g., proteins) is typically carried out in Cartesian coordinates, despite the fact that Cartesians are very inefficient for this purpose, due principally to the fairly large degree of coupling between the coordinates. Suitably choosen internal coordinates (stretches, bends, torsions) are much more efficient, at least as far as the number of optimization cycles required for convergence is concerned, with reductions of an order of magnitude and more compared to the corresponding Cartesian optimization [1-3]. Unfortunately, the transformation of geometries, gradients (and possibly Hessian matrices) - which are initially computed as Cartesian quantities - into internal coordinates (and in the case of the new geometry, back again to Cartesians) becomes increasingly more expensive as the system becomes larger, formally scaling as $O(N^3)$. Additionally, the memory requirement also increases; for example, the full Cartesian Hessian (or its inverse) for a system with 100 atoms requires less than 1 MB of double-word storage, for 10,000 atoms this has ballooned to a prohibitive 7.2 GB, clearly requiring a change of algorithm.

There are two parts to the efficient geometry optimization of very large molecules in internal coordinates: (1) Cut down the CPU time required for the coordinate transformations; (2) Develop an optimization algorithm which makes efficient use of in-core memory. We have combined these two steps to produce an efficient, large-molecule optimizer using internal coordinates.

BEST AVAILABLE COPY

There are several possible internal coordinates that one could choose for an efficient optimization, and we have developed large-molecule algorithms that use natural internal [4], delocalized internal [5] or redundant primitive internal coordinates [6]. As system size increases however, it is clear that only the latter remain generally viable. There are often problems generating a full set of natural internals, especially for systems containing complex topologies (such as fused rings and cages); such problems do not occur with delocalized internal coordinates, but large storage and memory demands set limits on the size of system that can be investigated. We already have efficient, near-linear, sparse-Cholesky transformation algorithms using primitive internals, and these form the basis of our large-molecule optimizer [4].

Originally we had hoped to use the Z-matrix backtransformation that we developed for delocalized internal coordinates [5] to speed up the transformation of a new geometry in internal coordinates back to Cartesians; unfortunately, despite several attempts, we were not able to successfully transplant this technique to either natural or primitive internals. Consequently, we are using a sparse-Cholesky backtransformation, which has been completely rewritten.

The optimization algorithm itself is currently based on the well known BFGS procedure [7]. This involves a direct update of the inverse Hessian based on current and previous gradients and the current displacement (step). For very large molecules (10,000+ atoms), the full Hessian matrix (which could have dimension around, say, 90,000, depending on the number of primitives) is too large to store and manipulate, and we have developed a limited memory BFGS algorithm which generates the inverse Hessian *in situ* (it actually forms the result of the inverse Hessian times the gradient) from an initial diagonal estimate and a (small) number of stored previous gradient and displacement vectors. Once the limit on the number of stored vectors is reached (e.g., 50), older vectors are simply replaced. For 90,000 primitives, this reduces the Hessian storage requirement from almost 64 GB to just 72 MB, a factor of 900. The algorithm that determines the actual optimization step requires storage for just a few vectors.

2. Internal Coordinate Dynamics

We have developed a fully-consistent Newtonian dynamics method using either natural [1] or delocalized [3] *internal* coordinates. Internal coordinates are particularly advantageous in low-temperature simulations, as the rigid degrees of freedom (stretches and most angle deformations) can be held fixed (or artificially slowed down) and only the low-frequency modes participate in the dynamics. As pointed out by Scheraga [8], this is also physically more accurate, since the rigid modes are not thermodynamically active in the real (quantum) system. SHAKE [9] and similar methods can be used to constrain bond distances but they do not work well for bond angles or rigid torsions, and they are much more awkward than our formulation.

The main obstacle to using internal coordinates in Newtonian dynamics was the extremely complex form of previous Lagrangian formulations [8,10,11]. We have overcome this by using Cartesian velocities to express the generalized centrifugal (Coriolis) forces. The resulting equations of motion are very simple:

$$d^2\mathbf{q}_i/dt^2 = \mathbf{B}^i\mathbf{M}^{-1}\mathbf{f} + \mathbf{v}^T\mathbf{C}^i\mathbf{v}$$

In this equation, $\mathbf{q}_i$ is an internal coordinate, *t* is the time, $\mathbf{B}^i$ is the *i*-th row of the Wilson **B** matrix relating internal to Cartesian coordinates to first order, **M** is the mass matrix, **f** is the vector of *Cartesian* forces, **v** is the vector of Cartesian velocities, and $\mathbf{C}^i$ is the second-order transformation matrix between internal and Cartesion coordinates according to

$$\Delta\mathbf{q}_i = \mathbf{B}^i\Delta\mathbf{x} + (1/2)\,\Delta\mathbf{x}^T\mathbf{C}^i\Delta\mathbf{x} + \ldots.$$

In our current formalism, **C** is obtained by central-differences on the B matrix.

In our first implementation [12], we had a minor problem with energy conservation, which was less accurate than in a Cartesian dynamics run using the same potential surface and initial conditions. This problem was traced to the propagation of Cartesian velocities. In order to avoid back transforming both the coordinates and the velocities, we propagated the Cartesian velocities independently. Numerical errors, accumulating slowly during the dynamics run, led to a slight inconsistency between the Cartesian velocities and the internal coordinates, and limited most runs to a relatively small number (~200,000) time steps. Our new implementation compares the internal velocities with the transformed Cartesian ones, and corrects the latter if the difference exceeds a small threshold. This has eliminated the problem.

As outlined above, a major factor in developing an internal coordinate dynamics algorithm is to impose geometrical constraints, i.e., to freeze the rigid degrees of freedom. This is best handled in delocalized internal coordinates, so that the very powerful constraint techniques already available in geometry optimization [13] can be transferred directly to dynamics. So far all our applications have involved unconstrained dynamics in natural internals, and we have only very recently modified our dynamics code to handle delocalized internals. We know exactly how to impose constraints during a delocalized internal coordinate dynamics run, and we hope to obtain further funding from the AFOSR to implement constrained dynamics, which we believe will constitute a significant advance.

We have also implemented a preliminary version of Fock matrix dynamics. This method combines the advantages of Car-Parrinello molecular dynamics (CPMD) with Born-Oppenheimer dynamics (BOMD). In CPMD, the wavefunction is not optimized but propagated using a fictitious mass. As a result it is stays close to, but is not identical with, the optimized wavefunction for the current geometry. The advantage of this procedure is that it requires only a single Fock matrix evaluation, instead of ~10 evaluations needed to determine the optimized wavefunction in BOMD. Note that there are violations of the Born-Oppenheimer approximation in the exact description but these are much smaller than the deviations introduced in CPMD. CPMD requires smaller time steps than BOMD.

Fock matrix dynamics exploits the fact that the individual Fock matrix elements in BOMD must be analytical functions of time, and thus can be extrapolated to the next time step. Fig. 1 shows the change in some Fock matrix elements during a few hundred steps in a typical molecular dynamics simulation. The curves appear exceptionally smooth. By using a suitable extrapolation polynomial, the optimized wavefunction can be determined by carrying out one full and 1-2 differential Fock matrix builds, instead of about ~10 steps which are needed if one uses simply the (renormalized) molecular orbitals from the previous time step. It approaches the efficiency of CPMD but the system stays on the Born-Oppenheimer potential surfaces. This eliminates some artefacts and permits longer time steps.

3. Correlation Energy in Large Molecules

Under this byline we have worked principally on two essentially independent topics, making excellent progress on each. Early on in the award period, we developed a very efficient canonical MP2 algorithm [14], which could routinely handle 1000+ basis functions on a standard PC; this algorithm was subsequently parallelized [15], and by making use of the aggregate disk storage capacity over the separate nodes, routinely allows calculations on systems with 2000+ basis functions. Currently, we are completing a canonical MP2 gradient code along similar lines.

More recently, we began developing a fast DFT code based on a plane wave expansion of standard Gaussian basis sets. Our approach, which we have termed the Fourier-Transform Coulomb (FTC) method [16,17], shares many of the goals of the Gaussian Augmented Plane Wave (GAPW) approach of Parrinello and coworkers [18], but is quite different technically. In particular, we aim at – and have achieved – much greater precision than is typical in methods using plane wave basis sets. Currently we have a successful (and parallel) first implementation for DFT energies which is up to 5-6 times faster than our standard all-integral code. The computation of the Coulomb term itself is over an order of magnitude faster.

3.1. Second-order Moller-Plesset Theory

3.1.a. An Efficient Canonical MP2 Program

The closed-shell MP2 energy can be written as [19]

$$E_{MP2} = \Sigma_{i>j}\, e_{ij} = \Sigma_{i\geq j}\, \Sigma_{a,b}\, (ai|bj)[2(ai|bj) - (bi|aj)]/(\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b) \qquad (1)$$

where i and j denote doubly occupied molecular orbitals (MOs), a and b denote virtual (unoccupied) MOs, and the ε are the corresponding orbital energies. The (ai|bj) are the two-electron repulsion integrals (in the usual Mulliken notation) over molecular orbitals. Virtually all of the computational work in calculating the canonical MP2 energy is associated with the evaluation of the atomic (AO) integrals $(\mu\nu|\lambda\sigma)$, and their transformation into the MO basis (ai|bj). Most existing algorithms carry out this transformation one index at a time, resulting in a formal fifth-order scaling with basis set size. Parallelization is also usually done over a single index, giving rise to a cubic scaling memory requirement. This often necessitates multiple passes through the integrals, as only a limited number of electrons can be correlated at any one time, diminishing the efficiency of the algorithm.

The Serial MP2 Algorithm

Our serial MP2 algorithm is based on the Saebo-Almlöf direct-integral transformation [20]. AO integrals are transformed via two half-transformations, involving first the two occupied MOs and then the two virtuals. If the integral $(\mu\nu|\lambda\sigma)$ is considered as the (ν,σ) element of a generalized exchange matrix $\mathbf{X}^{\mu\lambda}$, then the two half-transformations can be formulated as

$$(\mu i|\lambda j) = \mathbf{Y}^{\mu\lambda}_{ij} = \Sigma_{\nu,\sigma}\, C^{\dagger}_{\nu i}\, \mathbf{X}^{\mu\lambda}_{\nu\sigma}\, C_{\sigma j} = (\mathbf{C}^{\dagger}_{o}\, \mathbf{X}^{\mu\lambda}\, \mathbf{C}_{o})_{ij} \qquad (2a)$$

$$(ai|bj) = \mathbf{Z}^{ij}_{ab} = \Sigma_{\mu,\lambda}\, C^{\dagger}_{\mu a}\, \mathbf{Y}^{ij}_{\mu\lambda}\, C_{\lambda b} = (\mathbf{C}^{\dagger}_{v}\, \mathbf{Y}^{ij}\, \mathbf{C}_{v})_{ab} \qquad (2b)$$

Here $\mathbf{C}_o$ and $\mathbf{C}_v$ are the occupied and virtual parts of the SCF coefficient matrix. The disadvantage of this approach is that the 8-fold permutational symmetry of the AO integrals cannot be fully utilized, increasing the integral computation burden fourfold. This is probably why this it has not been seriously considered earlier. However, two important advantages of the Saebo-Almlöf technique more than compensate for the increased integral evaluation time in larger calculations. First, the fast memory demand grows only quadratically with the basis sets size, allowing very large calculations without multiple passes. Second, it allows the use of highly efficient dense matrix multiplication routines. The formal scaling of the integral transformation is dominated by the first quarter transformation, i.e., the first matrix multiplication in Eq. (2a). However, prescreening techniques, borrowed from the local MP2 method, allow the neglect of most elements of $\mathbf{X}^{\mu\lambda}$ in large molecules. In order to use efficient dense matrix multiplication methods, the matrices **X** are compacted, i.e., their zero rows and columns are eliminated. As each $\mathbf{X}^{\mu\lambda}$ is calculated, it is immediately transformed and the matrices $\mathbf{Y}^{\mu\lambda}$ are written to disk in compressed format. For the second half-transformation, the $\mathbf{Y}^{\mu\lambda}$ (which contain all indices i, j for a given μ,λ pair) have to be reordered into $\mathbf{Y}^{ij}$ (which contain all indices μ,λ for a given i,j pair). This is essentially the transposition of a very large matrix, and is accomplished via a standard Yoshimine bin sort [21]. The sorted bin files are then read back for each i,j pair, transformed via eq. (2b), and each pair's contribution to the correlation energy, e_{ij}, is computed and summed.

The entire scheme is straightforward to implement, provided there is enough memory to store a (potentially) complete $\mathbf{X}^{\mu\lambda}$ exchange matrix in core at one time, and enough disk space to hold all possible (compressed) $\mathbf{Y}^{\mu\lambda}$ matrices. Although the memory demand for the first half-transformation is only $O(N^2)$ in principle (where N is the number of basis functions), efficiency demands that AO integrals are calculated in batches over whole shells. This requires s^2N^2 double-words of fast memory, where s is the maximum shell size. This is a potential bottleneck for basis sets containing high angular momentum basis functions and its removal (currenly underway) will significantly increase the number of basis functions our MP2 program can handle.

A key to the efficiency of our MP2 program is prescreening of the AO integrals (based on the Schwarz inequality [22] and discussed in detail in ref. 14), and the compacting of the AO exchange matrices $\mathbf{X}^{\mu\lambda}$ which allows the use of highly efficient dense matrix multiplication routines. We compute only those AO integrals that make a contribution above an appropriate threshold (default 10^{-9}) to the pair correlation coefficients. Because of the efficiency of the prescreening, the second half transformation is often as important computationally as the first in our program. Symmetry can easily be utilized in the first half-transformation (this is more difficult in the second) by calculating only those integrals ($\mu\nu|\lambda\sigma$) which have symmetry-unique *shell* pairs M,L where $\mu \in M$, $\lambda \in L$.

The Parallel MP2 Algorithm

Using message passing (e.g., PVM or MPI), parallelization of the first part of the serial algorithm is straightforward; one simply loops over the M,L shells, sending each shell pair to an appropriate slave. At the end of the first half-transformation, each slave node will contain one or more half-transformed files containing the half-transformed $\mathbf{Y}^{\mu\lambda}$ matrices for whichever μ,λ pairs were computed on that slave.

The second half-transformation is done in essentially the same way, i.e., divide the i,j pairs equally among the slaves and transform each $\mathbf{Y}^{ij}$ matrix on the slave it is assigned to. Unfortunately, each $\mathbf{Y}^{\mu\lambda}$ matrix contains *all* i,j pairs, and so if we did a straightforward bin sort on each slave separately, we would have all possible i,j bin pairs on *every* slave. Consequently, we have modified the bin sort part of the algorithm. Our parallel bin sort starts by spawning a *second* process on each existing slave node, a "bin write" (or "bin listen") process. Whenever, during the bin sort, a particular bin on a given slave for a given i,j pair is full, instead of writing it to disk on that *same* slave, it is instead sent to the "bin write" process running on the slave the i,j pair is assigned to. The sort process knows in advance which i,j pairs should be sent to which slaves. The "bin write" process on the appropriate slave then writes the bin to its own local disk. At the end of sort all the "bin write" processes are killed, and each slave will have one or more sorted half-transformed integral files containing all μ,λ pairs for a subset of the i,j pairs.

The final half-transformation is done on each slave, and involves only the subset of the total number of i,j pairs that are on that slave. Each slave then computes a partial pair-correlation energy sum, and the partial sums are sent back to the master for the final summation to give the full MP2 correlation energy.

Both the serial and parallel algorithms are highly efficient, and the parallel algorithm scales well with the number of processors. Full details of both algorithms are described in references 14 and 15. Table 1, extracted from ref. 15, presents timings for single-point frozen-core MP2 energies on chlorophyll, taxol and tetraphenolporphine (containing up to 137 atoms and 1860 basis functions). These are fairly typical examples of the type of system that can be handled routinely with our MP2 algorithm. Calculations were performed on a 6-node Linux cluster, using 1.2 GHz Athlon processors, 1 GB memory and 50 GB scratch storage per node; on more modern machines (e.g., using 2.4 GHz Xeon processors) reported timings would be reduced by factors of between 2 and 3.

Table 1 Timings (min) for Single-point MP2 Energies for Chlorophyll a, Taxol and Tetraphenylporphine on a 6-node, 1.2 GHz Linux cluster

Molecule	Chlorophyll a	Taxol	Tetraphenylporphine
# atoms	137	113	78
symmetry	C_1	C_1	D_{2h}[c]
basis	VDZP	6-311G**	6-311G(2df,2pd)
# basis funcs.	1266	1422	1860
t_{SCF}	143	255	143
t_1[a]	179	411	155
$t_{sort} + t_2$[b]	681	814	470
t_{MP2}	878	1250	636
E_{SCF}	-2914.428668	-2912.569429	-1901.925096
E_{MP2}	-2923.570216	-2922.186323	-1909.687507

[a] elapsed time (minutes) for first half-transformation
[b] elapsed time (minutes) for bin sort + second half-transformation
[c] all four phenyl groups were perpendicular to the plane of the porphyrin ring

3.1.b. Local MP2

In spite of the outstanding efficiency of our new canonical MP2 program, it still retains, at least in the second half transformation, the steep scaling of the traditional MP2 method, both with the number of atoms and with the number of basis functions per atom. The cause of this – rather unphysical – scaling is the use of delocalized canonical molecular orbitals. Local correlation methods [23,24,25], originally introduced by one of us [26], are free of this defect. In particular, the latest local correlation implementations of Werner and coworkers [25] show excellent absolute timings and near-linear scaling with the number of atoms at constant basis set quality. Nevertheless, the local correlation method is not free of problems. Its results differ somewhat from the canonical results. This introduces a new model chemistry, which is undesirable: traditional quantum chemistry already suffers from too many methods and basis sets, making comparisons difficult. In the past, we have argued that difference between canonical and local results is due, at least partly, to the elimination of intramolecular basis set superposition effects [23]. However, in delocalized systems, there is a small but genuine error. Moreover, methods relying on localization can lead to artefacts when the localization changes suddenly, e.g., during the course of a chemical reaction.

An alternative to the usual local correlation methods, in particular to the Pulay-Saebo method, is to retain essentially *all* contributions which contribute to the correlation energy. This is the philosophy of Ayala and Scuseria [27] and also our new local MP2 program [28]. Local and canonical MP2 energies are identical if all significant contributions are included. Although more expensive than the original Saebo-Pulay method, the method still shows near-linear scaling with system size, c.f. Table 2. We use in this program an integral transformation technique in which two indices are transformed simultaneously. This method, first suggested by Taylor [29], has an unfavorable O(N^6) formal scaling, versus O(N^5) in the usual transformation. However, the sparsity of the integral list can be used more efficiently, ultimately resulting in better performance for very large systems. More details can be found in ref. 28.

Table 2 Scaling of the various steps for local MP2/6-31G* energy calculations on a series of polyglycines[a,b]

Number of	**(glycine)$_{10}$**	**(glycine)$_{15}$**	**(glycine)$_{20}$**	**Scale1**	**Scale2**
basis functions	638	948	1258		
correlated orbitals	114	169	224		
correlated pairs	441	656	871		
local dimension	243	257	264		
configurations	2.5×10^7	4.0×10^7	5.6×10^7	1.18	1.17
AO integrals	1.9×10^9	3.1×10^9	4.4×10^9	1.23	1.22
Transformed ints	4.3×10^7	7.2×10^7	10.1×10^7	1.29	1.18
Memory/MW	240	410	590	1.35	1.29
Disk storage	440	730	1000	1.28	1.11

[a] See ref. 28 for the molecular geometries

[b] Basis uses spherical harmonic (5-component) d functions. The number of contracted basis functions, correlated orbitals, correlated pairs, the average local dimension of the correlation space, the number of AO integrals evaluated, and the number of transformed integrals is given along with virtual memory demand (in Megawords; 1 MW=8 Mbyte), and disk storage (in Mbytes). *Scale1* of a quantity Q is $\log(Q_{15}/Q_{10})/\log(N_{15}/N_{20})$ where Q_{15} and Q_{10} are the values in the (glycine)$_{15}$ and (glycine)$_{10}$ calculations, respectively and N_{15} and N_{10} denote the number of contracted basis functions. *Scale2* is the analogous quantity for (glycine)$_{20}$/(glycine)$_{15}$

3.1.c. Dual-Basis MP2

We have also implemented a dual-basis MP2 method [30], originally proposed by Jurgens-Lutovsky and Almlöf [31]. SCF and MP2 calculations have different basis set requirements, and the highly oscillatory functions (of higher angular momentum) needed to describe close-range dynamical correlation are not necessary at the SCF level. In the dual-basis method, these high angular momentum functions can be omitted at the SCF level and included only in the MP2. Local correlation methods eliminate the steep scaling of explicit correlation (in this case MP2) calculations with the number of atoms, but do not improve the scaling with the number of basis functions for the same system. The dual-basis method has a significant impact here, and can easily give savings of up to an order of magnitude and more over the same calculation using the full basis throughout, with only a marginal loss of accuracy. This is illustrated in Table 3 (taken from ref. 30), which reports energies and timings for a number of calculations on the water trimer. The dual basis calculation is over 9 times faster than the full aug-cc-pV5Z basis calculation, and has an error of only 15 μh. Our original serial dual-basis code [30] has recently been fully parallelized.

Table 3 SCF and MP2 energies and elapsed time (min.) for calculations on the water trimer (on a single 2.4 GHz Xeon processor)

Basis	**# basis funcs.**	**E_{SCF}**	**T_{SCF}**	**E_{Tot}**	**T_{Tot}**
aug-cc-pVTZ	276	-228.196708	25	-229.012934	45
aug-cc-pVQZ	516	-228.212679	469	-229.081399	709
aug-cc-pV5Z	861	-228.216499	2953	-229.106077	5060
aug-cc-pV5Z(T1)[a]/ aug-cc-pV5Z dual	345/861	-228.215440 -228.216456[b]	50 183	-229.106092	549

[a] T1 basis is a subset of the full aug-cc-pV5Z basis obtained by removing 1d, 2f and all g & h functions from O, and 1p, 2d and all f & g functions from H

[b] includes "MP1" correction to SCF energy

3.2. The Fourier-Transform Coulomb (FTC) Program

There are several DFT codes that are significantly faster than the traditional integral-based programs developed by quantum chemists. The main reason for the increase in speed is that these codes use alternative ways of computing the Coulomb term, avoiding the calculation of the four-center integrals. Examples are the RI-DFT method - an old favorite of DFT researchers [32] - which uses an expansion of the density in an intermediate basis set, and the pseudospectral method developed principally by Friesner and coworkers from about 1983 onwards [33]. Additionally, several widely used DFT programs are based either fully (Delley's DMol [34]) or partially (ADF from Barends and coworkers [35]) on numerical solution of the Poisson equation. However, these alternative approaches usually introduce some level of approximation in the Coulomb energy, and speed is gained at the expense of accuracy.

We have developed a fast *and* accurate DFT program based on a (partial) expansion of standard gaussian basis sets in plane waves [16,17]. Plane waves have several disadvantages in molecular quantum chemistry, but also a number of tremendous advantages - the principal one being the ease of evaluation of the Coulomb potential - and we believe that the latter outweigh the former. In the past, the most prominent plane wave applications have been Car-Parrinello molecular dynamics (CPMD) and solid-state DFT, both with pseudopotentials and at a limited level of accuracy. Recently, Parrinello and coworkers have introduced the Gaussian Augmented Plane Wave (GAPW) approach [18], to which our FTC method is similar in spirit although quite different technically, as we are aiming at a much greater level of accuracy.

The basic idea behind the FTC method is to expand as much of the original basis set as possible in terms of plane waves, compute the electronic density on a direct-space grid, transform to momentum space to compute the Coulomb potential, transform back to the real space grid, and determine the Fock matrix elements (in the space of the original gaussian basis set) by numerical quadrature.

In general, typical gaussian basis sets cannot be expanded *fully* in plane waves as basis functions with large exponents (those representing the core region) cannot be represented sufficiently accurately in a plane wave basis. Consequently, we partition the original gaussian basis functions into two classes depending on the exponent value; those with small exponents (which we term *diffuse,* d) and those with large exponents (which we term *compact,* c). The exponent cutoff depends somewhat on the angular momentum (i.e., *S, P, D* etc...) of the basis function, and also on the quality of the grid (i.e., the number of plane waves in the expansion) but is around 3.0 a_0^{-2}. Partitioning the basis in this way results in the following classes of integrals that need to be evaluated (using the Mulliken notation):

(1) <cc|cc>; (2) <cc|cd>; (3) <cd|cd>; (4) <cc|dd>; (5) <cd|dd>; (6) <dd|dd>

Compact basis functions cannot be properly expanded in plane waves and so we treat the first four integral types using a variant of our standard integral package, i.e., essentially in exactly the same way as in a "normal" SCF integral code. Integrals of type (6) can be fully handled in plane wave space, as can those of type (5) (because the high momentum components of the charge density <cd> do not interact with the diffuse charge density <dd>). In theory, integrals of type (4) can also be taken over into plane waves, but we have not yet done this.

The molecule is placed in a box sufficiently large to contain essentially all the electron density. For simplicity, the box can be considered as a cube of sides L but in the actual program it is a parallelepiped, adapted to the molecular dimensions. We introduce a standard rectangular grid in our box, whose grid density, d - the number of plane waves in one Cartesian direction per atomic unit - characterizes the plane wave basis. The grid spacing is $h = d^{-1}$ and the grid points range from -L/2, -L/2 + h, ... , L/2 - h, L/2 in each Cartesian direction. The efficiency of Fourier transform and plane wave methods derives from the fast Fourier transform (FFT), which allows almost effortless switching between the momentum and coordinate representations. For quantities which can be exactly represented by the plane wave basis, the two descriptions are isomorphic. Note that we have already presented a method for eliminating errors due to the presence of periodic images [16,36].

Evaluation of the Coulomb potential in plane waves is extremely fast, and so we want as many basis functions as possible to be partitioned into the plane wave part of our space. Normally integrals of types (5) and (6) will dominate, especially in larger basis sets containing high angular-momentum and diffuse functions.

Having partioned the basis set, the steps followed to compute the Coulomb Fock matrix elements during each SCF cycle are as follows:

1. For those integrals of types (1) through (4), determine the Fock matrix elements in the normal way, i.e., specifically compute the integrals and contract them with the appropriate density matrix elements. This is done in a fully direct manner, using essentially the same code as in our standard all-integral program.

2. Compute the Coulomb contribution arising from the <dd|dd> integrals. This involves the following steps (order with respect to system size shown in parentheses):

   - Calculation of the "diffuse" density on the real space grid O(N)
     i.e., at each grid point (**r**): $\rho(\mathbf{r}) = \sum \mathbf{d}_{\alpha\beta} g^d_\alpha(\mathbf{r}) g^d_\beta(\mathbf{r})$, **d** = density matrix
   - Fast Fourier transform (FFT) to momentum space O(NlogN)
   - Calculation of the potential in momentum space O(N)
   - Reverse FFT back to real space O(NlogN)
   - Computation of Fock matrix elements by numerical quadrature O(N)

3. Compute the Coulomb contribution arising from the <cd|dd> integrals. This involves essentially the same steps as in 2, above, except that the "mixed" density $\rho(\mathbf{r}) = \sum \mathbf{d}_{\alpha\beta} g^c_\alpha(\mathbf{r}) g^d_\beta(\mathbf{r})$ is constructed, where one of the indices (α) corresponds to a compact gaussian function. In practice several steps from 2 and 3 are combined in the interests of program efficiency.

As indicated in 2, the scaling of the various steps is either O(N) or O(NlogN) for the

Fourier transform steps. In practice, the FFT steps are very fast; we use the excellent FFTW package of Frigo and Johnson [37]. The rectangular nature of our grid allows for very efficient screening and precomputation of many quantities.

The exchange-correlation part of the calculation is currently handled in the same way as in our integral-driven DFT code, i.e., by numerical quadrature over spherical, atom-centered grids, using techniques pioneered in this context by Becke [38]. We have already improved this part of the code (see, e.g., [39]), but we plan a further overhaul with a noticeably different quadrature allowing us to use much of the grid already generated and in place for the Coulomb term.

Table 4 presents a comparison of the job times for single-point DFT energies (using several different functionals) between our standard all-electron SCF code and the new FTC code using the 6-311G(2df,2pd) basis. All jobs were run in parallel on four 2.4 GHz PIV Xeon processors. As can be seen, the FTC code is significantly faster overall for all systems, by a factor of 3.7 for aspirin (the smallest molecule) rising to 6.0 for taxol. (There is a slight falloff for chlorophyll a, as other parts of the calculation, principally the diagonalization, now take proportionally longer.) Looking at the Coulomb contribution alone (the $T_{2\text{-el}}$ row in the all-integral calculation and the sum of the $T_{2\text{-el}}$ and T_{PW} rows in the FTC calculation), the factors are even greater, ranging from 5.7 for aspirin up to 10.8 for chlorophyll a.

Unlike some approaches, such as the continuous fast multipole method developed by Gill and Head-Gordon [40], the FTC method gives savings even with small molecules, such as aspirin (which would offer no savings at all with the fast multipole method as it is simply too small). The scaling, both with increasing system size at "constant" basis set size, and increasing basis set size at constant system size, is very favorable, and in the latter case – very important but often ignored – the scaling is genuinely linear, an achievement unequalled by any other non-plane wave approach. Furthermore, the accuracy is very high, with differences in energy between the FTC and all-integral codes in the μh range. For aspirin and yohimbine the *total* error is 1 μh or less, while for taxol and chlorophyll a the total error of a few tens of μh represents an error of less than 0.2 μh per atom.

Table 4 Timings (min) for Single-point DFT Energies for Aspirin, Yohimbine, Taxol and Chlorophyll a on a 4-node, 2.4 GHz Linux cluster

	Aspirin	**Yohimbine**	**Taxol**	**Chlorophyll a**
Formula	$C_9H_8O_4$	$C_{21}H_{26}N_2O_3$	$C_{47}H_{51}NO_{14}$	$C_{55}H_{72}N_4O_5Mg$
# atoms	21	52	113	137
Density functional	BLYP	BLYP	BVWN	OLYP
# basis funcs.	555	1275	2860	3309
T_{DFT}[a]	3.9	23.4	108.5	97.6
T_{misc}[b]	0.4	3.4	35.9	68.8
$T_{2\text{-}el}$[c]	33.2	239.9	1678	1539
$T_{classical}$[d]	37.5	266.7	1823	1705
$E_{classical}$	-648.719612	-1150.970847	-2952.295540	-2934.349474
T_{PW}[e]	3.1	9.1	24.1	29.4
$T_{2\text{-}el}$[f]	2.7	16.7	136.0	112.6
T_{FTC}[g]	10.1	52.6	304.5	308.4
E_{FTC}	-648.719612	-1150.970848	-2952.295565	-2934.349448

[a] elapsed time for calculation of exchange-correlation (DFT) energy

[b] elapsed time for all other steps (mainly Fock-matrix diagonalization)

[c] elapsed time for classical 2-el integral contribution to Fock matrix

[d] total elapsed time to compute SCF energy using classical all-integral algorithm

[e] elapsed time for all plane wave manipulations in FTC algorithm

[f] elapsed time for remaining classical integral contribution in FTC algorithm

[g] total elapsed time to compute SCF energy using FTC algorithm

3.3. Construction of Computer Clusters

To carry out the large-scale computations, we have constructed two connected computer clusters. We have used the experience gained with our older (1997) cluster, constructed with NSF funding. The cluster constructed in the first phase (2000) consists of five dual-processor nodes (800 MHz Pentium III, 512 MB of ECC memory, fast Ethernet), and two dual-processor nodes with larger memory (1 GB) and disk storage (in one case 180 GB of *striped* high-speed disks). This hardware was state of the art at its time. We jave later added two high-speed single-processor nodes (1.7 GHz Pentium IV and 1.33 GHz Athlon), for a total of 16 processors. It is running under the latest Linux operating system, and is very stable. Its performance has exceeded our expectations. In the second phase, we added 5 dual-processor and 4 single-processor large-memory (2 GB) nodes. All nodes in the second phase use 2.4 GHz Pentium 4 processors, and are connected by 1 Gigabit Ethernet network. For long-term stability, the cluster is powered by two 5 kVA uninterruptible power supplies. The total raw computing power of the current cluster (30 processors) is about 50 Gigaflops/s.

3.4. Miscellaneous

In addition to the above, we have also carried out a number of studies in related areas, principally involving density functional theory.

We have investigated the recently defined OLYP and O3LYP functionals [41,42] of Handy and Cohen, and their general applicability for studying organic reactions [43] and systems involving first-row transition metals [44]. The OLYP functional is especially promising – it was found to outperform the popular B3LYP functional for predicting bond lengths, heats of reaction and barrier heights for organic molecules [43] – and being a "pure" density functional (i.e., a non-hybrid, having no Hartree-Fock exchange), it can be used advantageously with our new fast DFT FTC code.

During the course of our investigations of the OLYP/O3LYP functionals [44], it became apparent that the standard 6-31G* basis set, recently extended to first-row transition metals by Rassolov and coworkers [45], performed poorly for many transition metals, particularly those towards the end of the series (Co, Ni, and especially Cu). This was tracked down to the lack of a sufficiently diffuse outer d-function in the original 6-31G* basis, and we have developed an improved 6-31G* basis, which we have termed m6-31G* [46], which provides results of consistent quality across the entire first-row transition metal series, and far better than the original basis for the higher members.

We have also completed several NMR applications projects, including two collaborations with other groups. A full list of all papers published and pending supported by this grant, and in which Air Force support has been acknowledged, is given below.

Papers Published in the Grant Period Acknowledging AFOSR Support

1. S. Saebo and P. Pulay, *A Low-Scaling Method for Second-Order Moller-Plesset Calculations*, J. Chem. Phys. **115** (2001) 3975
2. P. Pulay, S. Saebo, and K. Wolinski, *Efficient Calculation of Canonical MP2 Energies*, Chem. Phys. Lett. **344** (2001) 543
3. B. Wang, U. Fleischer, J. F. Hinton, and P. Pulay, *Accurate Prediction of Proton Chemical Shifts: I. Substituted Aromatic Hydrocarbons,* J. Comput. Chem. **22** (2001) 1887
4. L. Füsti-Molnár and P. Pulay, *Accurate Molecular Integrals and Energies Using Combined Plane Wave and Gaussian Basis Sets In Molecular Electronic Structure Theory,* J. Chem. Phys. **116** (2002) 7795
5. P. Pulay and B. Paizs, *Newtonian Molecular Dynamics In General Curvilinear Internal Coordinates*, Chem. Phys. Lett. **353** (2002) 400
6. B. Wang, J. F. Hinton, and P. Pulay, *Accurate Prediction of Proton Chemical Shifts: II. Peptide analogues,* J. Comput. Chem. **23** (2002) 492
7. P. Pulay, J. Baker and K. Wolinski, Reply to comments on 'Efficient calculation of canonical MP2 energies' by A. Kohn and C. Hättig, Chem. Phys. Lett. **358** (2002) 354
8. J. Baker and P. Pulay, *An Efficient Parallel Algorithm for the Calculation of Canonical MP2 Energies*, J. Comput. Chem. **23** (2002) 1150
9. J. Baker and P. Pulay, *Assessment of the Handy-Cohen Optimized Exchange Density Functional for Organic Reactions*, J. Chem. Phys. **117** (2002) 1441
10. B. Wang, M. Miskolzie, George Kotovych, and P. Pulay, *Backbone Structure Confirmation and Side Chain Conformational Refinement of a Bradykinin Mimic, BKM-824, by Comparing Calculated ^{1}H, ^{13}C and ^{19}F Chemical Shifts with Experiment,* J. Biomol. Struct. Dyn. **20** (2002) 71
11. L. Füsti-Molnár and P. Pulay, *The Fourier Transform Coulomb Method: Efficient and Accurate Calculation of the Coulomb Operator in a Gaussian Basis*, J. Chem. Phys. **117** (2002) 7827

12. H. V. Brand, R. L. Rabie, D. J. Funk, I. Diaz-Acosta, P. Pulay and T. K. Lippert, *Theoretical and Experimental Study of the Vibrational Spectra of the α, β, and δ Phases of Octahydro-1,3,5,7-tetranitro-1,3,5,7-tetrazocine (HMX)*,
J. Phys. Chem. B **106** (2002) 10594
13. A. V. Mitin, J. Baker, K. Wolinski and P. Pulay, *Parallel Stored Integral and Semi-Direct Hartree-Fock and Density Functional Energies with Data Compression*, J. Comput. Chem. **24** (2003) 154
14. A. V. Mitin, J. Baker and P. Pulay, *An Improved 6-31G* Basis Set for First-Row Transition Metals*, J. Chem. Phys. **118** (2003) 7775
15. K. Wolinski and P. Pulay, *Second-Order Møller-Plesset Calculations with Dual Basis Sets*, J. Chem. Phys. **118** (2003) 9497
16. I. Diaz-Acosta, J. Baker, J. F. Hinton and P. Pulay, *Calculated and Experimental Geometries and Infrared Spectra of Metal Tris-Acetylacetonates. Vibrational Spectroscopy as a Probe of Molecular Structure for Ionic Complexes. Part II. Jahn-Teller distorted complexes*
Spectrochim. Acta A **59** (2003) 363
17. J. Baker and P. Pulay, *Assessment of the OLYP and O3LYP Density Functionals for First-Row Transition Metals,* J. Comput. Chem. in press
18. B. Wang, J. F. Hinton and P. Pulay, *C-H···O Hydrogen Bond Between N-Methyl Maleimide and Dimethyl Sulfoxide: A Combined NMR and ab initio Study,*
J. Phys. Chem. A in press.
19. G. Magyarfalvi and P. Pulay, *Assessment of Density Functional Methods for NMR Shielding Calculations*, J. Chem. Phys. in press.

References

[1] G. Fogarasi, X. Zhou, P. W. Taylor and P. Pulay, *J. Am. Chem. Soc.* **114** (1992) 8191

[2] J. Baker, *J. Comput. Chem.* **9** (1993) 1085

[3] J. Baker, A. Kessi and B. Delley, *J. Chem. Phys.* **105** (1996) 192

[4] B. Paizs, G. Fogarasi and P. Pulay, *J. Chem. Phys.* **109** (1998) 6571

[5] J. Baker, D. Kinghorn and P. Pulay, *J. Chem. Phys.* **110** (1999) 4986

[6] B. Paizs, J. Baker, S. Suhai and P. Pulay, *J. Chem. Phys.* **113** (2000) 6566

[7] C. G. Broyden, *J. Inst. Math. Appl.* **6** (1970) 76; R. Fletcher, *Comput. J.* **13** (1970) 317; D. Goldfarb, *Math. Comput.* **24** (1970) 23; D. F. Shanno, *Math. Comput.* **24** (1970) 647

[8] K. D. Gibson and H. A. Scheraga, *J. Comput. Chem.* **11** (1990) 468

[9] J. P. Ryckaert, G. Ciccotti and H. J. Berendsen, *J. Comput. Phys.* **23** (1977) 327

[10] A. K. Mazur and R. A. Abagyan, *J. Biomol. Struct. Dyn.* **6** (1989) 815; 833

[11] A. K. Mazur, V. E. Dorofeev and R. A. Abagyan, *J. Comput. Phys.* **92** (1991) 261

[12] P. Pulay and B. Paizs, *Chem. Phys. Lett.* **353** (2002) 400

[13] J. Baker, *J. Comput. Chem.* **18** (1997) 1079

[14] S. Saebo, P. Pulay and K. Wolinski, *Chem. Phys. Lett.* **344** (2001) 543

[15] J. Baker and P. Pulay, *J. Comput. Chem.* **23** (2002) 1150

[16] L. Füsti-Molnar and P. Pulay, *J. Chem. Phys.* **116** (2002) 7795

[17] L. Füsti-Molnar and P. Pulay, *J. Chem. Phys.* **117** (2002) 7827

[18] M. Krack and M. Parrinello, *Phys. Chem. Chem. Phys.* **2** (2000) 2105

[19] P. Pulay, S. Saebo and W. Meyer, *J. Chem. Phys.* **81** (1984) 1901

[20] S. Saebo and J. Almlöf, *Chem. Phys. Lett.* **154** (1989) 83

[21] M. Yoshimine, *J. Comput. Chem.* **11** (1973) 333

[22] M. Abramowitz and I. A. Stegun, eds. *Handbook of Mathematical Functions* (Dover, New York, 1972)

[23] S. Saebo and P. Pulay, *Annu. Rev. Phys. Chem.* **44** (1993) 213

[24] C. Ochsenfeld, C. A. White and M. Head-Gordon, *J. Chem. Phys.* **109** (1998) 1663

[25] M. Schütz and H.-J. Werner, *Chem. Phys. Lett.* **318** (2000) 370

[26] P. Pulay, *Chem. Phys. Lett.* **100** (1983) 151

[27] P. Y. Ayala and G. E. Scuseria, *J. Chem. Phys.* **110** (1999) 3660

[28] S. Saebo and P. Pulay, *J. Chem. Phys.* **115** (2001) 3975

[29] P. R. Taylor, *Int. J. Quantum Chem.* **31** (1987) 521

[30] K. Wolinski and P. Pulay, *J. Chem. Phys.* **118** (2003) 9497

[31] R. Jurgens-Lutovsky and J. Almlöf, *Chem. Phys. Lett.* **178** (1991) 451

[32] B. I. Dunlap, J. W. Connolly and J. R. Sabin, *J. Chem. Phys.* **71** (1979) 3396

[33] R. A. Friesner, *Chem. Phys. Lett.* **116** (1985) 39

[34] B. Delley, *J. Chem. Phys.* **92** (1990) 508

[35] W. Ravenek and E. J. Baerends, *J. Chem. Phys.* **81** (1984) 865

[36] G. J. Martyna and M. E. Tuckerman, *J. Chem. Phys.* **110** (1999) 2810

[37] M. Frigo and S. G. Johnson, *Proceedings of the 1998 ICASSP Conference*, Vol. 3, pp. 1381-1384

[38] A. Becke, *J. Chem. Phys.* **88** (1988) 2574

[39] A. V. Mitin, J. Baker and P. Pulay, *J. Comput. Chem.* **24** (2003) 154

[40] C. A. White, B. G. Johnson, P. M. W. Gill and M. Head-Gordon, *Chem. Phys. Lett.* **230** (1994) 8

[41] N. C. Handy and A. J. Cohen, *Mol. Phys.* **99** (2001) 403

[42] W.-M. Hoe, A. J. Cohen and N. C. Handy, *Chem. Phys. Lett.* **341** (2001) 319

[43] J. Baker and P. Pulay, *J. Chem. Phys.* **117** (2002) 1441

[44] J. Baker and P. Pulay, *J. Comput. Chem.* in press

[45] V. A. Rassolov, J. A. Pople, M. A. Ratner and T. L. Windus, *J. Chem. Phys.* **109** (1998) 1223

[46] A. V. Mitin, J. Baker and P. Pulay, *J. Chem. Phys.* **118** (2003) 7775